Efficient DNN-Powered Software with Fair Sparse Models

Xuanqi Gao
Xi'an Jiaotong University
Xi'an, China
gxq2000@stu.xjtu.edu.cn

Weipeng Jiang
Xi'an Jiaotong University
Xi'an, China
lenijwp@stu.xjtu.edu.cn

Juan Zhai
University of Massachusetts at Amherst
Amherst, USA
juanzhai@umass.edu

Shiqing Ma
University of Massachusetts at Amherst
Amherst, USA
shiqingma@umass.edu

Xiaoyu Zhang
Xi'an Jiaotong University
Xi'an, China
zxy0927@stu.xjtu.edu.cn

Chao Shen
Xi'an Jiaotong University
Xi'an, China
chaoshen@mail.xjtu.edu.cn

Abstract

With the emergence of the Software 3.0 era, there is a growing trend of compressing and integrating large models into software systems, with significant societal implications. Regrettably, in numerous instances, model compression techniques impact the fairness performance of these models and thus the ethical behavior of DNN-powered software. One of the most notable example is the Lottery Ticket Hypothesis (LTH), a prevailing model pruning approach. This paper demonstrates that fairness issue of LTH-based pruning arises from both its subnetwork selection and training procedures, highlighting the inadequacy of existing remedies. To address this, we propose a novel pruning framework, Ballot, which employs a novel conflict-detection-based subnetwork selection to find accurate and fair subnetworks, coupled with a refined training process to attain a high-performance model, thereby improving the fairness of DNN-powered software. By means of this procedure, Ballot improves the fairness of pruning by 38.00%, 33.91%, 17.96%, and 35.82% compared to state-of-the-art baselines, namely Magnitude Pruning, Standard LTH, SafeCompress, and FairScratch respectively, based on our evaluation of five popular datasets and three widely used models. Our code is available at https://anonymous.4open.science/r/Ballot-506E.

CCS Concepts

• **Software and its engineering → Software design engineering**.

Keywords

fairness, deep neural network, pruning

ACM Reference Format:
Xuanqi Gao, Weipeng Jiang, Juan Zhai, Shiqing Ma, Xiaoyu Zhang, and Chao Shen. 2024. Efficient DNN-Powered Software with Fair Sparse Models. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '24), September 16–20, 2024, Vienna, Austria.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3650212.3680336

1 Introduction

We envisage that the Software 3.0 era, characterized by software powered by large-scale models, will pave the way for numerous potential applications, such as artificial intelligence generated content (AIGC) and autonomous driving, which are poised to exert substantial influence on societal transformations [76, 98]. The magnitude of AI software has notably surged, primarily driven by the escalating size of deep neural network models [63]. For instance, state-of-the-art computer vision models now encompass over 15 billion parameters, while large language models like GPT-3 exceed 175 billion parameters, demanding nearly 1TB of storage exclusively for the model itself [18].

However, the deployment of large-scale AI software introduces several challenges. These large models require substantial memory and storage resources during deployment, presenting difficulties in running them on resource-constrained devices like edge devices, smartphones, or wearables [36]. Furthermore, the size of these models often leads to slow inference times due to the sheer number of parameters and computations involved, which can be particularly problematic in time-critical applications such as real-time object detection or autonomous driving [55]. Additionally, when AI models are deployed on cloud servers and accessed remotely by client devices, the transfer of large model files over limited bandwidth can result in high latency and increased data consumption [88]. In addition to these deployment difficulties, the resource-intensive nature of large AI applications poses further challenges. The computational demands of these models significantly consume energy, which becomes a critical concern for battery-operated devices or data centers striving for energy efficiency [58]. Moreover, the carbon footprint associated with the training and deployment of large models can be substantial, contributing to environmental concerns [74].

To address these challenges, researchers employ model compression techniques, which aim to reduce the size and complexity of AI models while retaining their performance. Various model compression techniques have been introduced, encompassing model pruning[25, 41, 52, 59, 71], model quantization[44, 66], and knowledge distillation[11, 12, 42]. Notably, the model pruning algorithm stands out as one of the most widely adopted methods in this domain[56, 63]. Pruning involves the selective removal of unnecessary parameters, connections, or entire neurons, leading to a more streamlined and efficient model. As a state-of-the-art method, the Lottery Ticket Hypothesis (LTH) has garnered significant attention and extensive research efforts [25]. The hypothesis posits the existence of a winning ticket, namely a properly pruned subnetwork combined with the original weight initialization, which can achieve competitive performance comparable to that of the original dense network. This discovery underscores the immense potential for efficient training and network design in the realm of deep learning.

Unfortunately, among the various research on the LTH and also our experiment results (see §2.3), LTH-based pruning methods suffer from model bias problem [61]. Through our analysis, we found that the bias problem is caused by both ticket selection and ticket training. The process of ticket selection entails the elimination of redundant neurons, resembling high-dimensional feature selection for images, and potentially introducing bias. Fairness and accuracy can exhibit conflicts during ticket selection, with prevailing LTH-based pruning methods prioritizing accuracy, resulting in tickets that lack fairness. Even after obtaining a ticket that balances accuracy and fairness, there is no assurance that the ticket can be trained to reach its theoretical performance upper limit. The training process faces potential issues of overfitting and underfitting, impacting the model's ability to generalize and make accurate and fair predictions. Current efforts to enhance the LTH-based pruning primarily concentrate on improving its efficiency and accuracy performance, neglecting its fairness concerns.

Inspired by ethics-aware software engineering [10, 19, 28], we present Ballot, a novel fairness-aware pruning framework by revised winning ticket finding and training for deep neural networks. The key idea of Ballot is that it follows the best practice of software engineering by first observing and analyzing the root cause of the bias problem. Concretely, it performs a conflict detection between the fairness optimization direction and the accuracy optimization direction of the model during the model training process to identify neurons with the highest degree of conflict. These neurons are more likely to induce the model to optimize towards greater variance among distinct subgroups during training, leading to biased model behavior and subsequently giving rise to instances of discriminatory behavior in the software. Each neuron is assigned a score representing its importance in training optimization conflicts, and a corresponding mask is generated to remove these neurons, resulting in the acquisition of an accurate and fair ticket. To ensure that the ticket can eventually be trained into high-performance models, we also refine the training procedure. This involves adapting the learning rate based on the training condition to facilitate the model's learning of richer features. Furthermore, we verify the fairness performance of the model upon the completion of training. If the fairness performance is inferior to that of the original model, we reload the weights from earlier training rounds (the specific number of rounds predetermined by empirical considerations) and iteratively retrain the model until further performance improvement is unattainable. This refined training approach is designed to maximize the ticket's performance potential.

Ballot has been implemented as a self-contained toolkit. Our experiments on CIFAR-100, TinyImageNet, and CelebA datasets show that Ballot can effectively mitigate the pruning fairness problem while maintaining accuracy, compared with existing pruning methods [26, 38, 81, 99].

Our contribution can be summarized as follows:

- We propose a novel fairness-aware DNN pruning framework. It leverages conflict-detection-based mask generation to find fair and accurate tickets, and training refinement to achieve optimal performance.
- We develop a prototype Ballot based on the proposed idea, and evaluate it with three mainstream datasets. On average, Ballot improves fairness by 38.00%, 33.91%, 17.96%, and 35.82% compared to state-of-the-art baselines, namely Magnitude Pruning, Standard LTH, SafeCompress, and FairScratch, outperforming all baselines in terms of fairness.
- Our implementation, configurations and collected datasets are available at [1].

2 Background and Motivation

2.1 DNN Software and DNN Model Deployment

The continuous advancements in deep neural network (DNN) research and the availability of powerful computing resources have contributed to the widespread adoption of DNNs in various applications, making DNN-powered software a key driver of innovation in the field of software engineering [62]. DNN-powered software refers to applications and systems that leverage the capabilities of DNNs for various tasks, which are often developed with deep learning frameworks and libraries such as PyTorch [3], TensorFlow [5], and Keras [34]. However, as the capabilities of deep learning models gradually increase, their scale also becomes larger and larger, naturally taking up a lot of storage space [65]. The deployment DNN software encounters complexities due to the diverse software landscape and deployment demands on various mobile platforms. Confronted with resource limitations, such as computational power, storage capacity, and battery life, especially in embedded devices like mobile devices, the imperative arises for mobile models to adhere to stringent conditions encompassing reduced model size, diminished computational complexity, and minimal battery power consumption [56]. Consequently, a challenge is raised for DNN-powered software developers: how can a DNN-powered software be effectively deployed to meet performance requisites on platforms characterized by constrained resources? The compression of DNN models emerges as an indispensable operation in the deployment of DNN-powered software.

Numerous studies emphasize the requisite step of model compression for deploying DNN models across diverse platforms [56, 65]. Predominantly employed techniques for DNN model compression encompass model quantization [66, 80], model pruning [59, 71], and knowledge distillation [12, 42]. Among these, the model pruning is widely favored and commonly utilized, due to its capability to directly reduce the number of computational operations involved, fundamentally alleviating computation and memory pressures [54, 55]. Popular AI software development frameworks such as PyTorch [85], Tensorflow [82] and PaddlePaddle [68], all integrate pruning-based model compression toolkits.

Model pruning is a kind of a prominent and effective compression technique in the field of large-scale AI software deployment. The motivation behind model pruning stems from the recognition that many deep learning models, especially those with an over-parameterized nature, contain numerous redundant or less critical parameters. By selectively removing unimportant components or parameters from a model, this approach achieves a remarkable reduction in model size, rendering it more lightweight and computationally efficient as well as not significantly sacrificing model performance. In a DNN, the architecture consists of layers of interconnected nodes (neurons), and each connection between nodes is associated with a weight. These weights, along with biases, are the parameters of the model, and they are collectively represented by the symbol θ. A DNN model $f(\theta)$ is a function that takes input data and produces output based on these parameters. More formally, model pruning aims to produce a lightweight model $f(\theta^*)$ satisfying the following conditions:

$$\frac{|\theta^*|}{|\theta|} \leq \Omega \tag{1}$$

$$Acc(f(\theta)) - Acc(f(\theta^*)) \leq \epsilon \tag{2}$$

The $Acc(\cdot)$ measures accuracy, which is the most frequently utilized performance metric. $|\theta^*|$ and $|\theta|$ are the numbers of parameters, ϵ and Ω are the tolerable decline of performance and the sparsity metric respectively. Sparsity metric Ω declares the level of pruning, for instance, the $\Omega = 0.05$ requires the removal of 95% of the parameters, i.e., the pruning rate with 0.95. In practice, the goal is to maximize the accuracy of the pruned model, i.e. achieve a smaller ϵ while adhering to a fixed sparsity metric Ω to satisfy deployment requirements. Pruned models can even achieve improved robustness and generalization, as the reduction of parameters helps alleviate issues such as overfitting [46, 54].

2.2 Fairness Problem in Pruning

While model pruning strikes a trade-off between accuracy and model size, recent research take attention to potential fairness concerns. For example, Paganini points out that pruning should not only focus on overall performance metrics but should also consider performance differences across subgroups and individuals, i.e., the fairness of model [69]. In machine learning, model fairness refers to the ethical and unbiased treatment of individuals or groups throughout the development, deployment, and utilization of machine learning models. Consequently, the definition of fairness is also divided into two categories: individual fairness and group fairness. Blakeney et al. find that the bias of the model grew progressively as the pruning rate increased [15]. For a pruning technique to be considered fair, the bias degree of the small model after pruning should not significantly increase compared to the metrics of the original model before pruning. Formally, for an original model $f(\theta)$ and a pruned model $f(\theta^*)$, the fairness criterion can be defined as:

$$d(f(\theta^*)) - d(f(\theta)) \leq \delta \tag{3}$$

where $d(\cdot)$ computes the bias degree (i.e. higher values represent lower levels of fairness), and δ is the tolerance. As previously mentioned, the definition of fairness is categorized into individual fairness and group fairness. This paper specifically concentrates on group fairness. For datasets with sensitive features, fairness metrics such as demographic parity (DP) [22] and equal opportunity (EO) [39] are commonly employed to assess whether the model exhibits discrimination against distinct groups. Since vision and natural language processing tasks usually have neither explicit sensitivity characteristics nor explicit favorable/unfavorable treatment, we utilize class-wise variance (CWV) [84] and maximum class-wise discrepancy (MCD) [84] to measure variations in the model's performance across different groups, i.e. to measure the bias degree.

These metrics are formally defined below:

Class-wise Variance (CWV). Given a dataset with C classes and a given model, the accuracy on the subdataset belonging to c_{th} class is denoted as acc_c.

$$\text{CWV} = \frac{1}{C}\sum_{c=1}^{C}(acc_c - \bar{acc})^2, \ \bar{acc} = \frac{1}{C}\sum_{c=1}^{C} acc_c \tag{4}$$

Maximum Class-wise Discrepancy (MCD). For a model, given the maximum and minimum class accuracy acc^+ and acc^-, MCD can be defined as:

$$\text{MCD} = acc^+ - acc^- \tag{5}$$

Intuitively, CWV represents the average discrepancy, while MCD assesses the extreme discrepancy among classes to estimate the fairness of a model. Larger values of these metrics indicate a higher degree of bias in the model. For example, for a software that includes a face recognition module, each class corresponds to one user, and if there are some users whose recognition accuracy is significantly lower than the average or other users (resulting in a large CWV or MCD), it implies unfair treatment, as users with equivalent qualifications receive disparate service quality. While our primary assessment of model fairness revolves around these two metrics, our approach readily extends to other fairness metrics such as DP and EO, contingent on the user defining advantageous outcomes based on the scenario. We select these metrics because they are well-established and commonly used metrics [13, 31, 81] that are generally applicable to various tasks and datasets including both vision and NLP tasks discussed in this paper.

2.3 Lottery Ticket Hypothesis

A recently proposed technique known as the Lottery Tickets Hypothesis (LTH) has emerged as a rapidly growing method in the field of model pruning, which focuses on sparse trainable subnetworks within fully dense networks [25]. The LTH posits the presence of a sparse subnetwork within a randomly initialized dense network, which can achieve test accuracy comparable to that of the original dense network within at most the same number of iterations during independent training. This sparse subnetwork is known as the "winning ticket". More specifically, for a randomly initialized dense neural network $f(\theta_0)$ parameterized by $\theta_0 \in \mathbb{R}^{|\theta_0|}$, LTH suggests searching a mask $m \in (\{0,1\}^{|\theta_0|})$, i.e., the winning ticket is the sparse subnetwork $f(\theta_0 \odot m)$, where $\odot$ denotes the element-wise product and $|\theta_0|$ denotes the numbers of parameters. To find a winning ticket, it is common practice to first train the initialized network for some epochs and compute a mask based on the behaviors of the trained model. LTH-based pruning usually follows the *train-prune-retrain* pipeline as shown in Figure 4. Given a dense neural network $f(\theta_0)$ initialized by the parameter $\theta_0 \in \mathbb{R}^{|\theta_0|}$ and the sparsity metric Ω, the process of finding a winning ticket and perform pruning can be expressed as:

i) Train the network $f(\theta_0)$ for E epochs to get a well-trained model $f(\theta)$.
ii) Remove parameters in θ which satisfied the sparsity metric Ω,by creating a mask $m \in (\{0,1\}^{|\theta_0|})$. Reset the remaining parameters to their initial values in θ_0, creating the winning ticket $f(\theta_0 \odot m)$.
iii) Retrained the $f(\theta_0 \odot m)$ to get final pruned model $f(\theta^*)$.

Ideally, the performance of $f(\theta^*)$ is expected to surpass that of $f(\theta)$ within a limited number of training epochs, not exceeding E. However, in a standard LTH setup [25], a winning ticket can only be found when the learning rate is very small, and the performance advantage will become smaller when the learning rate is large. How to find lotteries that actually win is still a problem worth exploring, but this does not detract from the practical value of LTH-based pruning to select high-value subnetworks. A series of efforts were devoted to finding the most promising ticket, which can achieve higher performance[60].

Despite the remarkable success of LTH-based pruning in achieving high performance, it is crucial to recognize that it still grapples with the challenge of fairness. To demonstrate this matter, we pruned the widely utilized open-source model ResNet50 Gender Classifier [2] using PyTorch's official LTH pruning algorithm package [3]. Subsequently, we assessed the fairness and accuracy of the original model on the CelebA dataset [57] concerning its sparser versions obtained through LTH-based pruning, with pruning ratios set at 0.9, 0.95, and 0.99, respectively. The accuracy, CWV, and MCD are recorded for each model, and the results are presented in Figure 1. The bars in the figure are pattern-coded, with diagonal-hatching representing the original model, and crosshatching, dotted pattern, and plus sign pattern corresponding to pruning rates of 0.9, 0.95, and 0.99 (i.e. sparsity of 0.10, 0.05 and 0.01), respectively. As depicted in the figure, while not leading to a significant decrease in model accuracy, an increase in model sparsity results in a significant reduction in fairness performance.

This observation highlights that the LTH approach, being primarily focused on identifying a winning ticket that can retain its capacity toward test accuracy throughout training, tends to overlook potential fairness issues. Consequently, a pertinent question arises: *Is it feasible to identify a winning ticket that excels in both accuracy and fairness concurrently?*

3 System Design

3.1 Problem Statement

In this paper, we aim to develop an automated model pruning framework that concurrently addresses both pruning rate and fairness considerations. For a given initialized complex network $f(\theta_0)$ and training epochs E, the $f(\theta_0)$ can be normally trained into a model $f(\theta)$ with E epochs. We focus on how to select a ticket to satisfy the specified sparsity metric Ω, and retrain this ticket into a final sparse model $f(\theta^*)$ by a training method T. The $f(\theta^*)$ should attain a high level of fairness, while fulfilling the availability requirement stated by the sparsity metric Ω and accuracy specifications ϵ as Equation 1 and Equation 2. Formally, our objective is:

$$\min_{m,T} \left[d(f(\theta^*)) - d(f(\theta)) \right] \tag{6}$$
$$\text{s.t.} \quad \frac{|\theta^*|}{|\theta|} \le \Omega \quad \text{Acc}(f(\theta)) - \text{Acc}(f(\theta^*)) \le \epsilon$$

where the $d(\cdot) \in \{CWV, MCD\}$ is a measure of bias degree referred to §2.2, which represents a higher fairness with lower indicator value. The $|\theta|$ and $|\theta^*|$ calculate the number of parameters, while $Acc(\cdot)$ calculates the accuracy of the model on the test dataset. Note the pruning rate is $(1 - \Omega)$. Our objective is to enhance the fairness of the pruning model as much as possible while preserving its performance (i.e., accuracy). This involves finding a balance between accuracy and fairness, making trade-offs when necessary.

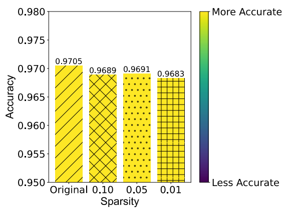
3.2 Root Cause Analysis

In pursuit of an accurate and fair ticket, it is essential to consider the trade-off involved in optimizing both aspects. We believe that trade-off arises at two primary levels: first, in determining what components of the model should be removed during pruning (called *ticket selection*); and second, in devising an appropriate training strategy for the model after the removal (called *ticket training*). Below, we conduct an exhaustive analysis of the factors contributing to fairness issues in both ticket selection and ticket training.

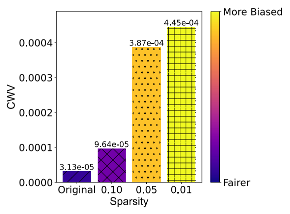
3.2.1 *Ticket Selection.*

Fairness and accuracy may conflict in the search for tickets. For DNN models, the ticket selection and pruning process involves removing redundant neurons. Typically, high-dimensional features extracted from samples are stored in neurons as combinations. Thus, ticket selection is equivalent to feature selection on the dataset, which represents a form of high-dimensional feature selection for a sample. However, this selection process may introduce bias. As previously discussed, LTH methods predominantly prioritize accuracy, potentially leading to the identification of tickets that lack fairness. Such tickets may resort to taking shortcuts instead of genuinely learning the intended solution. For instance, in a face recognition software deployed in a predominantly white neighborhood, a ticket may only use skin color as the determinant of whether a visitor is a local resident, resulting in reduced generalization and fairness [32]. Conversely, exclusively focusing on fairness during ticket identification may result in a decline in accuracy. Intuitively, if a face recognition software blocks out all features related to skin color to be fair, its face recognition performance is likely to suffer considerably. Hence, a trade-off must be carefully considered throughout the process of ticket selection to strike a balance between fairness and accuracy.

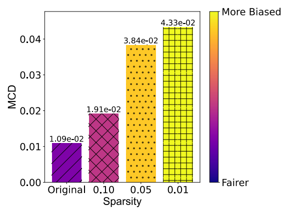
We study the association of neurons to features by employing an interpretable AI technique known as GradCAM [75]. The GradCAM heatmap visually represents the regions in an image that the deep learning model deemed most influential in making its prediction, wherein "hot" regions indicate areas of significant influence on predicting the object's class membership. The primary objective of using the GradCAM heatmap is to visualize the regions of the input image that the model's attention is directed towards under varying masks, shedding light on the model's feature focus during its decision-making process. The experiment is based on the ResNet-50 model, which was trained on the PubFig dataset [49]. Figure 2 shows how the neuron mask affects the model's selection



(a) Accuracy



(b) CWV (larger value, higher degree of bias)



(c) MCD (larger value, higher degree of bias)

Figure 1: Performance of ResNet50 Gender Classifier and its sparse versions after LTH-based pruning.

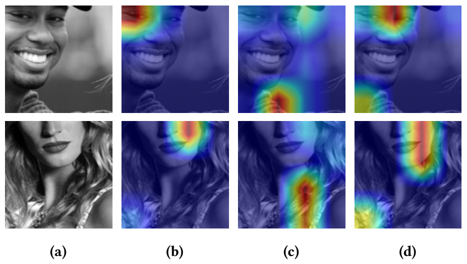
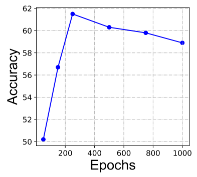


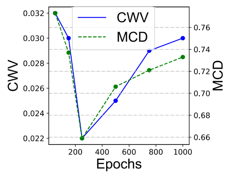
Figure 2: GradCAM heatmap for different models. (a) is the original image; (b)-(d) are the results of the original model, the model after random mask application and the model after Ballot pruning.

of features. In the figure, regions highlighted in orange depict areas where the model exhibits high attention, while regions colored in blue represent areas with lower model attention. Note that (a) represents the original image, while (b), (c), and (d) correspond to the saliency maps of the original model, the model after random mask application, and the model after Ballot pruning, respectively. Obviously, the region of interest of the model undergoes a shift as the neurons involved in decision-making are different. The region of interest of the model after random mask depicted in (c) exhibits a significant shift compared to the original model (b), diverting its focus from facial features to other attributes that are unrelated to face classification but may carry discriminatory information (e.g., hair color, clothing, etc.). In contrast, the model after Ballot pruning appears to maintain a closer resemblance to the original model, which likely indicates that the model prioritizes the region containing features essential for face classification. This observation underscores the substantial influence that the judicious selection of neurons can have on the fairness of the model. We further corroborate this through subsequent empirical experiments (see §4.4).

3.2.2 Ticket Training. Even upon obtaining a ticket that strikes a balance between accuracy and fairness, there is no guarantee of successfully "claiming a prize", that is, training a model that attains the upper limit of its theoretical performance. There are two potential issues related to the training process. First, if the model excessively focuses on learning the features of existing training samples, it may treat some of these specific features as if they are universally representative of all potential samples, leading to overfitting. In such cases, the model becomes too tailored to the training data and may not generalize well to new, unseen samples, thus introducing bias. Conversely, if the model's learning capacity is too limited, it may not adequately capture the general features of the samples. This situation, known as underfitting, results in a failure to learn critical patterns from the data, leading to poor performance on both training and unseen samples. Achieving an



(a) Epochs-accuracy



(b) Epochs-CWV&MCD

Figure 3: Average model performance on CIFAR-100 dataset for different training epochs setup.

optimal balance between overfitting and underfitting is essential for obtaining a model that generalizes well and demonstrates accurate predictions on new and diverse samples.

We conduct a series of training experiments on the CIFAR-100 dataset, utilizing the same ticket (model prototype) acquired through standard LTH pruning with a pruning rate of 0.95. The experiments involve training the model for 50 and 150 epochs (underfitting model), 250 epochs (normally fitting model), and 500, 750, and 1000 epochs (overfitting model). Throughout all experiments, a uniform learning rate of 0.03 is employed. Then we evaluate the accuracy and fairness of these trained models. Figure 3 compares the accuracy, CWV, and MCD of these models. As we can see, the normal model exhibits the most favorable performance in both fairness and accuracy, whereas the underfitting model demonstrates the poorest performance. This highlights that, despite employing the same ticket, the training process significantly influences the final performance of the retrained model.

3.3 System Overview

Through our analysis, it becomes evident that existing methods encounter challenges in finding tickets that simultaneously achieve both accuracy and fairness while successfully training a model. In response, our proposed framework, Ballot, introduces novel *conflict-detection-based mask generation* and *training refinement* techniques to obtain accurate and fair sparse models. We first locate the root cause, namely neurons exhibiting conflicts between fairness and accuracy during optimization, and then generating masks that block out these neurons to get a sparse model and train it by a comprehensive strategy to improve its fairness performance. The primary objective of mask generation is to ensure that the model focuses on the most relevant features during the learning process. The training refinement is pivotal in ensuring that the model not only captures essential features but also achieves equitable outcomes across different groups or classes.

Figure 4 presents the workflow of Ballot. Following the conventional LTH approach, we commence by training randomly initialized models while recording the gradient values of each neuron

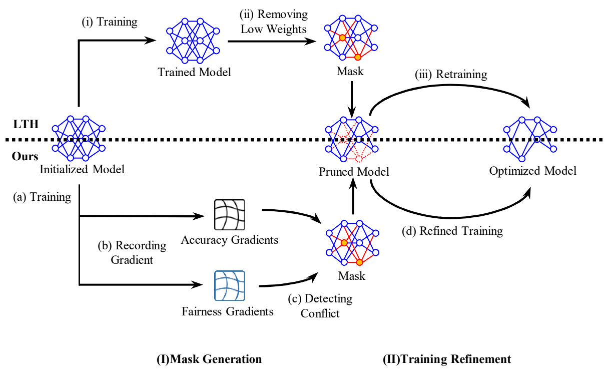


Figure 4: Overview of Ballot.

during each round of model training with respect to accuracy loss and fairness loss. Subsequently, we employ these gradient values to conduct an analysis and identify neurons that present conflicts between fairness and accuracy optimization objectives. Next, we generate masks corresponding to these identified neurons and proceed to prune them to address the conflicts. Finally, we undertake a training refinement step on the sparse model to achieve an optimized model. We dynamically adjust the learning rate during training to enhance the model's acquisition of more nuanced features, and if the trained model's fairness lags behind the original model, we iteratively retrain it with weights reloaded from earlier rounds until further improvement is unattainable.

The overall algorithm of Ballot is presented in Algorithm 1, denoted as procedure *FixModel*. It takes a base model $f(\theta_0)$, the predetermined sparsity metric Ω and the number of training epochs E as inputs, and outputs an optimized model $f(\theta^*)$. Firstly, Ballot engages in a standard training process on the base model $f(\theta_0)$, recording the accuracy loss L_a and the fairness loss L_f for each epoch (line 2-7). This meta-data collection aims to acquire insights into the optimization directions concerning accuracy and fairness during training, facilitating subsequent conflict analysis. Next, Ballot utilizes this information to identify the mask corresponding to a ticket that satisfies both accuracy and fairness (line 8). Traditional LTH methods often prioritize accuracy, leading to subpar fairness performance. In contrast, Ballot guides the model to optimize towards the direction of conjugate optimization by analyzing the directions of fairness and accuracy optimization, thereby enhancing the likelihood of discovering a model that is both accurate and fair. Finally, Ballot employs training refinement (lines 9-11) to optimize the performance of the discovered tickets. Through this refined training, Ballot guarantees the preservation of model accuracy while concurrently striving to uphold fairness.

3.4 Mask Generation

To address this challenge mentioned in §3.2.1, we propose a novel mask generation approach termed *conflict-detection-based mask generation*. Specifically, we collect the gradients corresponding to the fairness and accuracy of each neuron during training the model in the normal way, and then locate neurons with optimization conflicts and generating masks to remove them.

Algorithm 1 Ballot Algorithm

```
Input: $f(\theta_0)$: base model with random initialized weight
Input: $\Omega$: sparsity metric
Input: $E$: number of training epochs
Output: $f(\theta^*)$: optimized model
 1: procedure FixModel
 2:     $L_a \leftarrow []$                                         ▷ list of model accuracy loss
 3:     $L_f \leftarrow []$                                         ▷ list of model fairness loss
 4:     for $i \leftarrow 0$ to $E$ do
 5:         $f(\theta_{i+1}) \leftarrow Training(f(\theta_i))$
 6:         $L_a[i] \leftarrow AccLoss(f(\theta_{i+1}))$
 7:         $L_f[i] \leftarrow FairLoss(f(\theta_{i+1}))$
 8:     $m \leftarrow GenerateMask(L_a, L_f)$
 9:     for $i \leftarrow 0$ to $E$ do
10:         $f(\theta_{i+1} \odot m) \leftarrow RefinedTraining(f(\theta_i \odot m))$
11:     $f(\theta^*) \leftarrow f(\theta_E \odot m)$
        return $f(\theta^*)$

Input: $L_a$: list of accuracy loss for each epoch
Input: $L_f$: list of fairness loss for each epoch
Input: $\gamma$: hyperparameter used to control the loss weight
Input: $\eta$: hyperparameter used to control the neuron sorting ratio
Output: $m$: mask corresponding to fair and accurate ticket
12: procedure GenerateMask
13:     $L_c \leftarrow []$                                         ▷ list of model weighted loss
14:     $count \leftarrow []$                      ▷ Record the number of conflicts for each neuron
15:     for $i \leftarrow 0$ to $|\theta|$ do
16:         $count[i] \leftarrow 0$
17:     for $i \leftarrow 0$ to $E$ do
18:         $L_c[i] \leftarrow L_a[i] + \gamma \times L_f[i]$
19:         $l \leftarrow Sort(L_c[i].neuronvalue)$
20:         for $j \leftarrow 0$ to $\eta \times |\theta|$ do
21:             $count[l[j].index] \leftarrow count[l[j].index] + 1$
22:     $countsorted \leftarrow Sort(count)$
23:     for $i \leftarrow 0$ to $\Omega \times |\theta|$ do
24:         $Append(m, countsorted[i].neuron)$
        return $m$
```

Prior to delving into our approach, let us first introduce the precise definitions of the accuracy loss function (denoted as l_a) and the fairness loss function (denoted as l_f) employed in our methodology. The l_a and l_f can be formulated as:

$$l_a = -\sum_{c=1}^{C} \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c} \tag{7}$$

$$l_f = -\sum_{c=1}^{C} w_c \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c} \tag{8}$$

In the context of the provided equations, C denotes the number of classes, x represents the input data, y denotes the target and w corresponds to the weight. Here, we set the weights as $\frac{1}{\hat{acc}_c}$, where $\hat{acc}_c$ denotes the accuracy of the previous iteration of the model. For each respective class, we take this value as an estimate of the current iteration's accuracy. The choice of weights follows an inverse proportionality to the accuracy achieved in the prior iteration for individual classes.

Initially, during the standard training process, we collect the gradient values of each neuron pertaining to both the fairness and accuracy loss functions for each training round (line 4-7). This information serves as the foundation for evaluating potential optimization conflicts. Then we compute the sum of these two values for every round, weighted by a certain percentage γ to ensure that both values are on the same order of magnitude (line 12-18). Here, the value of γ is predetermined based on the outcomes of hyperparameter experiments, as discussed in §4.5. The resulting sums are sorted to derive a reference value, denoted as conflict degree l, delineate the degree to which the optimization objectives of the two loss functions are in conflict during each respective round (line 19). Next, we find out the neurons with the highest degree of conflict and increment their respective count (line 20-21). By summing the count values of each neuron across all rounds, we identify the neurons with the highest number of conflict rounds (line 22-24). These neurons are then removed based on the sparsity metric to obtain the desired mask for our approach.

Through an examination of the metadata from the training process, we identify neurons whose optimization exhibited conflicts between the fairness and accuracy objectives. This analysis aid in the discovery of potential models that are more inclined to achieve fairness-accuracy conjugate optimization by eliminating these neurons. However, merely identifying such a sparse model prototype (ticket) is insufficient; it remains crucial to consider how to effectively train this prototype into a fully functional and high-performing model (claim a prize). The optimization and validation of ticket training are imperative.

3.5 Training Refinement

Upon finding a ticket, our task is only halfway completed. While this ticket holds the potential to serve as a fair and accurate model, careful training is crucial. Traditional LTH methods involve loading the ticket with the initial base model weights θ_0 and then retraining, akin to the initial training. However, such an approach lacks optimization steps and neglects performance validation of the trained model. Thus, there is a need for a more comprehensive training strategy that includes optimization techniques and robust performance validation to ensure the model attains its full potential.

To address this, we undertake a comparison of various training policies and optimize the training procedure employing a technique termed *training refinement*. In detail, we commence the training refinement process by reloading the initial weights into the pruned model. Subsequently, we conduct model training using a decreasing learning rate strategy to facilitate learning as many essential features as possible while avoiding overfitting. The objective is to strike a balance between adequately capturing necessary patterns in the data and ensuring the model's ability to generalize effectively to new, unseen samples. Ultimately, we validate the retrained model. If the model's fairness performance falls below that of the original model, we initiate a feedback loop by returning to the first step. During this process, instead of reloading the base model initialization weights, we reload the weights after completing a specified number (typically a small single or double digit number predetermined by empirical considerations, see §4.4) of training epochs and recommence the training procedure. This is because sparse subnetworks that remain stable to the optimization noise in the early stages of training are more likely to be winning tickets [27]. We iterate through the aforementioned steps until fairness or accuracy decreases above the threshold value ϵ or ceases to improve further. It increases the likelihood of identifying and training winning tickets to the desired levels of accuracy and fairness. By following the aforementioned process, we mitigate potential issues such as convergence failure or convergence to a local optimum, which may arise during the training process. Our approach endeavors to ensure that the previously obtained accurate and fair tickets can be effectively employed to train a high-performing model.

4 Evaluation

We aim to answer the following research questions through our experiments:
RQ1: Can Ballot effectively claim the winning ticket?
RQ2: How efficient does Ballot find and claim the winning ticket?
RQ3: How do mask generation and training policies affect the performance of Ballot?
RQ4: What is the impact of different configurable parameters in Ballot?

4.1 Setup

4.1.1 Software and Hardware. The prototype of Ballot is implemented on top of PyTorch 2.0. We conduct our experiments on a server with 64 cores Intel Xeon 2.90GHz CPU, 256 GB RAM, and 4 NVIDIA 3090 GPUs running the Ubuntu 16.04 operating system.

4.1.2 Datasets. We evaluate Ballot on five popular datasets: CIFAR-100 [4], TinyImageNet [50], CelebA [57], LFW [43], and Moji [16].

- **CIFAR-100** is a widely-used colored image dataset used for object recognition. It contains 60,000 32x32 color images in 100 classes, of which 50,000 are training images and the rest are test images.
- **TinyImageNet** is a subset of the ILSVRC2012 classification dataset. It consists of 100k colored images of 200 classes, and all images have been down-sampled to $64 * 64 * 3$ pixels.
- **CelebA** is a large-scale face attributes dataset with more than 200K celebrity images. The images in this dataset cover large pose variations and background clutter. We perform face gender classification task on this dataset.
- **LFW** is a collection of over 13,000 face photographs sourced from the web. Many groups are underrepresented in this dataset, such as children, babies, individuals over the age of 80, and women. Therefore, LFW is suitable for evaluating model performance in the data imbalance setting. We perform face gender classification task on this dataset.
- **Moji** contains more than 100,000 tweets written in either "Standard American English" or "African American English", annotated with positive or negative sentiment. We perform text sentiment classification task on this dataset.

4.1.3 Models. For a fair comparison with baseline methods, we reproduce the baselines and use the same models and setup on each method in the experiments. We train the 50-layer ResNet model (i.e., ResNet50) [40] and the VGG16 [78] with SGD and set the batch size to 32. In the training process, the learning rate starts from 0.1 and reduces to 1/10 of the previous learning rate after 100, 150 and 200 epochs (250 in total). When pruning models, we set the sparsity to 0.05, i.e., remove 95% of the parameters in the model. Random cropping and horizontal flip are adopted for data augmentation.

To assess the performance of the Ballot in natural language processing tasks, we conducted experiments on the BERT model [21]. We finetune the BERT model with AdamW, set the batch size to 32 and the learning rate to 5e-5.

4.1.4 Baselines. We compare Ballot with other state-of-the-art model pruning methods, including Magnitude Pruning [38], Standard LTH [26], SafeCompress [99], and FairScratch [81]. In experiments, We run the baseline methods with the default settings which are recommended in their papers and open-source code and record their performance.

Magnitude Pruning. Han et al. [38] trained a model to learn which neuronal connections are important, and then they pruned those unimportant ones and fine-tuned the target model. Their method can effectively reduce the number of parameters of the target model by an order of magnitude without affecting their accuracy.

Standard LTH. Frankle et al. [26] proposed the Lottery Ticket Hypothesis (i.e., Standard LTH in this paper) that a DNN contains subnetwork (e.g., the winning ticket) that is able to match the test accuracy of the original network after training. They found the winning ticket by masking the original model, thereby reducing the number of parameters of the model while ensuring accuracy.

SafeCompress. Zhu et al. [99] proposed SafeCompress, which is a test-driven sparse training framework for safe model compression. SafeCompress first prunes the big model to an initial sparse model, and then iteratively compresses the sparse model with various compression strategies. We reuse their open-source code by replacing the safety metric with fairness metrics.

FairScratch. Tang et al. [81] proposed a fairness learning framework for computer vision system without weight training. It tries to find a suitable mask in the initialized original model, whose corresponding subnetwork has a high performance without training.

First three works designed model pruning and compression methods from the perspective of model performance and security, which effectively enhance the accuracy and safety of pruned models. However, they all raise fairness issues. The reason is that none of them consider the impact of pruning on model bias and fairness, nor use any metrics as guidance. In fact, during the pruning process, they may prune some neurons that are strongly associated with fairness in order to ensure the accuracy of the model, thereby introducing bias into the model, which makes it difficult for the model to achieve high fairness. FairScratch, on the other hand, does not consider the increased difficulty of identifying winning tickets without weight training, particularly in scenarios with high compression rates, which limits its performance.

4.1.5 Evaluation metrics. We compare BALLOT with the baselines on five metrics: accuracy, precision, recall, CWV and MCD (see §2.2).

4.2 RQ1: Effectiveness in Finding and Claiming Winning Ticket

Experiment Design: To evaluate the effectiveness of BALLOT in finding and claiming in the winning ticket to fix the bias problems, we use the following three baseline pruning methods and BALLOT to conduct experiments on ResNet50, VGG16 and BERT on the five datasets, which are mentioned in §4.1. In the experiment, we fixed the training settings and dataset of the model and ran the training 10 times to ensure the reliability of the results and reduce the randomness. We compare the performance between BALLOT and baseline methods in terms of both utility and fairness.

Results: The comparison results are presented in Table 1. The first column shows two datasets and the second column lists four methods in experiments including BALLOT. We record the model performance of the original models that have not been pruned, which is represented as 'Origin' in the second column. The remaining columns list the performance of the pruned model, including accuracy (Acc.), precision (Precis.), recall, class-wise variance (CWV), and maximum class-wise discrepancy (MCD). The best results among different pruning methods are highlighted in bold.

Table 1: Effectiveness evaluation results.

Model	Dataset	Methods	Acc.	Precis.	Recall	CWV	MCD
ResNet50	CIFAR-100	Magnitude Pruning	0.5892	0.5923	0.5881	0.0262	0.7493
		Standard LTH	0.6145	0.6186	0.6160	0.0254	0.7056
		SafeCompress	0.6163	0.6149	0.6168	0.0254	0.6396
		FairScratch	0.5957	0.5993	0.5952	0.0276	0.6407
		BALLOT	**0.6449**	**0.6444**	**0.6436**	**0.0225**	**0.6243**
		Original Model	0.6390	0.6417	0.6390	0.0229	0.6134
	TinyImageNet	Magnitude Pruning	0.4136	0.4291	0.4296	0.1205	0.3565
		Standard LTH	0.4454	0.4268	0.4457	0.1193	0.3333
		SafeCompress	0.4460	**0.4657**	0.4436	0.1098	0.4242
		FairScratch	0.4133	0.4197	0.4153	0.1222	0.3436
		BALLOT	**0.4534**	0.4442	**0.4512**	**0.1073**	**0.2500**
		Original Model	0.4662	0.4377	0.4643	0.1173	0.2065
	CelebA	Magnitude Pruning	0.9528	0.9501	0.9524	4.89e-4	0.0404
		Standard LTH	0.9691	**0.9695**	0.9687	3.87e-4	0.0384
		SafeCompress	0.9526	0.9619	0.9527	1.37e-4	0.0292
		FairScratch	0.9644	0.9649	0.9645	3.54e-4	0.0396
		BALLOT	**0.9702**	0.9689	**0.9707**	**8.21e-5**	**0.0138**
		Original Model	0.9705	0.9698	0.9711	3.13e-5	0.0109
	LFW	Magnitude Pruning	0.9635	0.9628	0.9641	9.23e-4	0.0335
		Standard LTH	0.9700	0.9693	0.9706	7.30e-4	0.0315
		SafeCompress	0.9524	0.9517	0.9530	1.37e-4	0.0192
		FairScratch	0.9770	0.9763	0.9776	3.34e-4	0.0215
		BALLOT	**0.9850**	0.9843	**0.9856**	**5.54e-5**	**0.0121**
		Original Model	0.9865	0.9858	0.9871	3.59e-5	0.0123
VGG16	CIFAR-100	Magnitude Pruning	0.5899	0.5930	0.5888	0.0524	0.7979
		Standard LTH	0.6150	0.6191	0.6165	0.0508	0.7112
		SafeCompress	0.6168	0.6154	0.6173	0.0508	0.6784
		FairScratch	0.5962	0.5998	0.5957	0.0552	0.6821
		BALLOT	**0.6456**	**0.6451**	**0.6443**	**0.0450**	**0.6486**
		Original Model	0.6395	0.6422	0.6395	0.0458	0.6268
	TinyImageNet	Magnitude Pruning	0.4143	0.4298	0.4303	0.2410	0.3530
		Standard LTH	0.4461	0.4275	0.4464	0.2386	0.3669
		SafeCompress	0.4466	**0.4663**	0.4443	0.2196	0.4484
		FairScratch	0.4139	0.4203	0.4159	0.2444	0.3872
		BALLOT	**0.4539**	0.4448	**0.4518**	**0.2146**	**0.2605**
		Original Model	0.4667	0.4382	0.4648	0.2346	0.2130
	CelebA	Magnitude Pruning	0.9533	0.9506	0.9530	0.0098	0.0308
		Standard LTH	0.9696	**0.9700**	0.9692	0.0077	0.0318
		SafeCompress	0.9531	0.9624	0.9532	0.0027	0.0304
		FairScratch	0.9649	0.9654	0.9650	0.0031	0.0292
		BALLOT	**0.9707**	0.9694	**0.9712**	**0.0016**	**0.0276**
		Original Model	0.9710	0.9703	0.9716	0.0063	0.0218
	LFW	Magnitude Pruning	0.9642	0.9635	0.9648	6.30e-3	0.0470
		Standard LTH	0.9705	0.9698	0.9711	3.44e-3	0.0420
		SafeCompress	0.9529	0.9522	0.9535	5.61e-3	0.0264
		FairScratch	0.9775	0.9768	0.9781	3.32e-3	0.0230
		BALLOT	**0.9855**	**0.9848**	**0.9861**	**1.61e-4**	**0.0233**
		Original Model	0.9870	0.9863	0.9876	1.22e-4	0.0266
BERT	Moji	Magnitude Pruning	0.9188	0.9123	0.9142	0.4467	0.9984
		Standard LTH	0.9437	0.9472	0.9411	0.4469	0.9975
		SafeCompress	0.9336	0.9341	0.9318	0.3254	0.6966
		FairScratch	0.9430	0.9435	0.9412	0.2887	0.6536
		BALLOT	**0.9636**	**0.9647**	**0.9566**	**0.1254**	**0.3323**
		Original Model	0.9612	0.9523	0.9681	0.1168	0.3360

The experiment results demonstrate that the winning tickets exist and BALLOT can effectively find and claim these lottery tickets. On the one hand, BALLOT can effectively alleviate the fairness bias in model pruning and the ticket BALLOT found has better performance in fairness than other baselines. The last two columns of Table 1 show the fairness improvement of BALLOT on these datasets. The models pruned by BALLOT achieve the lowest CWV (indicates the highest fairness) among all models, which exceeds the Magnitude Pruning by 32.52%, 53.71%, and 71.92%; Standard LTH by 35.60%, 56.78%, and 71.91%; SafeCompress by 60.87%, 58.33%, and 61.49%; and FairScratch by 43.02%, 60.11%, and 56.57% on pruning ResNet50, VGG16, and BERT. On average, BALLOT improves CWV by 49.05%,

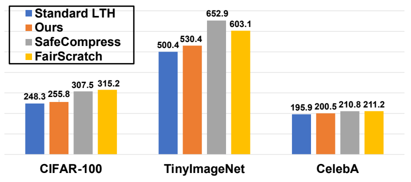


Figure 5: Time to prune and retrain a model.

54.76%, 60.23%, and 53.23% compared to baselines, namely Magnitude Pruning, Standard LTH, SafeCompress, and FairScratch. In terms of MCD, the Ballot also has advantages in alleviating the class-wise discrepancy over other baselines. MCD of Ballot exceeds the Magnitude Pruning by 43.81%, 44.79%, and 66.72%; Standard LTH by 48.25%, 46.88%, and 66.67%; SafeCompress by 57.81%, 52.78%, and 52.21%; and FairScratch by 46.20%, 49.18%, and 49.17% on pruning ResNet50, VGG16, and BERT. On average, Ballot improves MCD by 51.77%, 53.93%, 54.27%, and 48.18% compared to state-of-the-art baselines, namely Magnitude Pruning, Standard LTH, SafeCompress, and FairScratch. Although the MCD result of the model pruned by Ballot is slightly behind the original model, it has already significantly exceeded the four baselines.

On the other hand, Ballot effectively preserves and even improves the model accuracy, precision, and recall when pruning models and achieves the highest utility among all methods. From Table 1, we can observe that Ballot outperforms the state-of-the-art methods on accuracy, precision and recall when pruning models. Compared to the non-pruned models (i.e., original model), Magnitude Pruning degrades 8.51%, 8.75%, and 4.41%; Standard LTH degrades 3.03%, 3.25%, and 1.75%; SafeCompress degrades 8.79%, 9.05%, and 2.76%; and FairScratch degrades 1.30%, 1.45%, and 1.82% on pruning ResNet50, VGG16, and BERT respectively. In contrast, the degradation of model accuracy of Ballot is only 0.70% and 0.85% for ResNet50 and VGG on average. Ballot even improves the accuracy of the original model by 0.92% on the CIFAR-100 dataset for ResNet50 and 0.24% on the Moji dataset for BERT.

Analysis: From Table 1, we can have the following observations. First, compared with other baselines, Ballot can prune the given models and make them more effective (higher accuracy, precision, and recall) and fair (higher fairness performance). Secondly, compared with the original models, the model pruned by Ballot doesn't degrade the accuracy, and in some cases, the pruned models can achieve higher test accuracy. This phenomenon could be attributed to the model pruning process enabling a concentrated emphasis on pivotal neurons, representing combinations of essential features, as noted in prior studies [29, 89]. Therefore, in conclusion, Ballot can effectively find and claim the winning ticket which can take both utility and fairness into account. In other words, Ballot can fix the bias issues in model pruning and retraining and guarantee the utility and fairness of the pruned model.

4.3 RQ2: Efficiency in Finding Winning Ticket

Experiment Design: To evaluate the efficiency of Ballot in finding and claiming the winning ticket, we measure the time cost of Ballot and the baselines (i.e. Standard LTH, SafeCompress, and FairScratch) in performing a complete pruning and retraining process on these datasets. Since the Magnitude Pruning method does not contain the retraining process after pruning, which makes it difficult to make a fair comparison, we do not involve this method in the comparison. To avoid the effect of randomness, we perform 10 trials that use random training/test data splitting and compute and record the average time overhead of each method. The corresponding results and analysis are presented below.

Table 2: Comparison among different mask generation and training policies in Ballot. *w/o* is short for *without.*, and Ballot (*n*th) is short for Ballot rewinding to *n*th epoch.

Method		Acc.	Precis.	Recall	CWV	MCD
Mask Policy	Random	0.6383	0.6353	0.6379	0.0261	0.6538
	Magnitude	0.6401	0.6395	0.6410	0.0258	0.6712
	Ballot	**0.6449**	**0.6444**	**0.6436**	**0.0225**	**0.6243**
Training Policy	w/o LR decreasing	0.5922	0.5905	0.5924	0.0282	0.7512
	w/o rewind	0.6285	0.6237	0.6277	0.0235	0.6566
	Ballot (5th)	0.6411	**0.6448**	0.6423	0.0229	0.6339
	Ballot (10th)	**0.6449**	0.6444	**0.6436**	**0.0225**	**0.6243**
	Ballot (15th)	0.6425	0.6421	0.6431	0.0233	0.6301

Results: Figure 5 shows the time cost of each method and demonstrates the efficiency of Ballot in finding fair sparse model. The blue, orange, gray, and yellow bars represent the time cost of the four methods respectively. On average, on the CIFAR-100, TinyImageNet, and CelebA datasets, Ballot spends 255.8 minutes, 530.4 minutes, and 200.5 minutes, which are 16.80%, 18.76%, and 4.89% faster than the SafeCompress, and 18.85%, 12.05%, and 5.07% faster than the FairScratch. Compared with the Standard LTH method, Ballot takes slightly more time. The average execution time of Ballot is 3.02%, 6.00%, and 2.35% longer than the Standard LTH.

Analysis: As shown in Figure 5, compared to baselines, Ballot costs close or less time on pruning and retraining. Ballot spent significantly less time than the SafeCompress method on finding and claiming lottery tickets, and its time cost is close to that of the Standard LTH. Combined with the results in Table 1, we can see that Ballot achieves far better results than the baseline in terms of fairness and utility while costs similar or even less time, which demonstrates the efficiency of Ballot in finding fair sparse models.

4.4 RQ3: Impacts of Mask Generation and Training Policies

Experiment Design: Ballot leverages two core components, namely the *conflict-detection-based mask generation* and *training refinement*, to find and claim the winning tickets. In this section, we aim to demonstrate the effectiveness of these two components by comparing the performance of the pruned model under different mask generation and training policies. To this end, 1) for mask generation policies, we compare the results of finding winning tickets among the conflict-detection-based mask generation implemented in Ballot (see §3.4), random mask generation, and magnitude mask generation. The random policy randomly selects neurons to generate masks. The magnitude one removes those neurons with the smallest weights to generate a mask, which is applied in Standard LTH [26]. 2) for training policies, we compare the performance among Ballot rewinding to 5th epoch, Ballot rewinding to 10th epoch (default), Ballot rewinding to 15th epoch, Ballot without learning rate decreasing, and Ballot without rewind. The latter two are training policies that do not use learning rate decreasing or the rewind method, which are introduced in §3.5. We follow the experiment settings in §4.1 and conduct experiments on the

CIFAR-100 dataset. We record and compare the accuracy, precision, recall, CWV, and MCD of these models to observe how different mask and training policies affect their utility and fairness.

Results: Table 2 presents the results, where the best results are marked in bold. The first columns list different mask generation and training policies, and the following five columns show the results of the pruned models on accuracy, precision, recall, CWV, and MCD. Overall, Ballot achieve the comprehensively best fix results among the compared methods, which indicates that the conflict-detection-based mask generation and training refinement in Ballot can help to find fair sparse models.

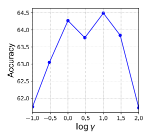
Impact of mask generation policy: The first three rows of Table 2 show the comparison results of mask generation methods, where the first line shows the results of the default method in Ballot. When Ballot applies the random mask generation instead of conflict-detection-based mask generation, the accuracy of pruned models decreases by 1.02%, and CWV and MCD increase by 15.87% and 4.73%. In addition, using the magnitude mask generation method instead of the one in Ballot also causes performance degradation. It decreases the accuracy by 0.74% and increases CWV and MCD by 14.67% and 7.51% respectively. It follows that the model obtained by BALLOT pruning is more effective in maintaining fairness, in line with our observations in §3.2.1.

Impact of training policy: The last five rows show the effect of different training policies. Compared to the default training refinement in Ballot, Ballot without learning rate decreasing and Ballot without rewind reduce the accuracy of pruned models by 8.17% and 2.54% and increase the CWV by 25.47% and 4.44%. We can observe that removing any method in the learning rate decreasing and the rewind will cause a significant degradation in the performance of Ballot. In the context of employing the rewind method, loading the weights from the 10th epoch yields better results compared to loading from the 5th and 15th epoch, with a 1.75% and 3.43% rise in CWV and 1.51% and 0.92% increase in MCD.

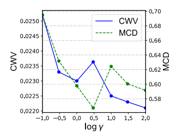
Analysis: The above results show that compared with the other optional methods, the conflict-detection-based mask generation and training refinement in Ballot achieves the highest utility and fairness, which can improve model fairness when ensuring effectiveness. In addition, both mask generation and training policy implemented in Ballot is helpful to improve the performance of the pruned model. It is noteworthy that Ballot rewinding to 10th epoch is slightly better over both 5th epoch and 15th epoch. This observation aligns with the theory proposed in [27], suggesting that the original network's weights are prone to stability against SGD noise after an initial training period. However, an extended training duration may result in weight solidification.

4.5 RQ4: Impacts of Configurable Parameters

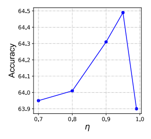
Experiment Design: Ballot leverages two hyperparameters γ and η in mask generation. The former is used to control the contribution of fairness loss L_f and accuracy loss L_a. The latter controls the neuron sorting ratio, which determines which neurons in each iteration will be considered to have the highest degree of conflict and increment the count. We conduct experiments to investigate and understand how different values of these configurable hyperparameters affect the performance of Ballot in finding and claiming the winning ticket. In the experiment, we run Ballot on the CIFAR-100 dataset with different $\log \gamma$ values from -1.0 to 2.0 and different η values from 0.7 to 0.99. To evaluate the effect of different values of configurable parameters, we record the accuracy, CWV, and MCD of the model pruned by Ballot with different parameter settings.



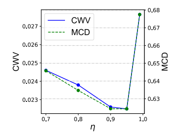
(a) γ-accuracy



(b) γ-CWV&MCD



(c) η-accuracy



(d) η-CWV&MCD

Figure 6: Effect of γ and η on model average performance.

Results: Figure 6 shows the comparison results of different values of the configurable parameters γ and η, where Figure 6(a) and Figure 6(c) presents how γ and η affect the accuracy of the pruned models in Ballot, and Figure 6(b) and Figure 6(d) show the effect of different values of γ and η on the fairness of the pruned model.

Impact of γ: Ballot uses the parameter γ to control the impact of the fairness loss in mask generation. A higher γ leads the pruned model to pay more attention to fairness. As shown in Figure 6(a) and Figure 6(b), when the value of $\log \gamma$ increases, the accuracy of the model pruned by Ballot first increases and then decreases, while the values of CWV and MCD generally show a downward trend. When $\log \gamma$ is 1.0, the accuracy of the pruned models reaches the highest value of 64.49%. It is noteworthy that at the two points of 0.5 and 1.0, there are abnormal increases in the values of CWV and MCD, and then their values continue to decline as the value of $\log \gamma$ increases. However, the increment brought by these outliers is relatively small, and the impact on the model fairness is not significant. Considering the trade-off between accuracy and fairness, Ballot selects 1.0 as the default value of $\log \gamma$ (i.e., γ is set to 10) to maximize the effectiveness on improve model fairness and utility.

Impact of η: η is used to determine which neurons will be regarded as conflicting neurons and counted. A higher η indicates that in mask generation of Ballot, the criteria for evaluating conflicting neurons are more strict. The experiment results in Figure 6(c) and Figure 6(d) show that the model performance of both utility and fairness first increase and then decreases as η rises, and Ballot performs best when η is set to 0.95. Therefore, we choose 0.95 as the default value of η.

Analysis: From Figure 6, we can observe that different values of γ and η have a significant impact on the performance of the pruned model. In terms of accuracy, as the values of γ and η increase, the model accuracy first increases and then decreases. Similarly, in terms of fairness, the increase of η first improves fairness and then worsens it. In addition, the increase of γ improves the model fairness in general. As a result, to effectively select the winning ticket and prune a sparse model toward both accuracy and fairness, we set the default value of γ to 10, and the value of η to 0.95 in Ballot.

Summarization. We have proved the advancement of BALLOT through the above experimental evaluations. Compared to the baseline, BALLOT effectively balances fairness and accuracy throughout the pruning process and efficiently identifies equitable subnetworks, thus expediting training. BALLOT requires a more intricate procedure than conventional pruning methods, resulting in higher training costs than direct pruning. However, this extra expense remains relatively modest, averaging no more than a 6% increase.

5 Threat to Validity

Ballot is evaluated on five mainstream datasets with ResNet50, VGG16, and BERT under the sparsity of 0.05, which may be limited. Additionally, the existence of several configurable parameters introduces some threats. While our experiments demonstrated promising pruning results, the effectiveness remains uncertain and challenging when applied to more complex and advanced models like transformer. To address those concerns, we have taken the step of open-sourcing the implementation of Ballot and providing comprehensive experimental details that encompass models before and after pruning, as well as training configurations. For reproduction purposes, all codes and data are available at [1].

6 Related Work and Discussion

Model Compression. Model compression aims to reduce the size and computational complexity of large neural network models. There are various model compression methods, such as network pruning, parameter quantization, and knowledge distillation.

Network pruning. Pruning removes redundant components without greatly affecting performance. Channel-based pruning removes entire feature maps from convolutional networks [71], while filter-based pruning removes specific filters [41, 52]. Unimportant connections are chosen based on criteria like weight magnitude [37, 38], gradients [64], and hessian statistics [51].

Parameter quantization. Quantization reduces network weights and activations from high to low precision. Quantization-aware training (QAT) simulates quantization errors during training [66]. Post-training quantization (PTQ) directly quantizes the pre-trained model without retraining, though it might cause more accuracy loss [44].

Knowledge distillation. Knowledge distillation uses a teacher-student architecture where a complex teacher network provides a simple student network with prior knowledge to match the teacher's performance. Various methods use different forms of prior knowledge for training student networks [12, 42].

The software engineering community has exhibited significant interest in model compression. Shi et al. compress pre-trained code models to 3MB for easy deployment [77]. Zhu et al. explore test-driven development for pruning using bi-objective optimization to balance performance and safety [99]. Quality assurance for the deployment of compression-based AI models is a subject of great concern. Xie et al. conduct differential testing to compare the behaviors of models before and after compression [90]. Additionally, Zhang et al. propose an ILP-based formal verification approach for quantized neural networks [97].

Fairness of ML. The issue of fairness in ML is gaining increasing prominence, in the areas of CV [83], NLP [9, 92] and especially critical automated decision-making systems like higher education [20], employment [33, 72, 87], and re-offense judgement [17, 67].

To address the concern above, fairness testing for ML systems is gaining attention in the software engineering community. AEQUITAS proposes a directed search for individual discriminatory instances [86]. Symbolic Generation (SG) integrates symbolic execution and local model explanation to craft individual discriminatory instances [7]. ExpGA uses the genetic algorithm to generate discriminatory instances [23]. Besides, ADF and EIDIG combine global search and local search to systematically explore the input space with the guidance of gradient [95, 96], and BREAM further extends ADF to black-box scenarios by introducing shadow model for approximate gradient guidance [45]. There are also concerns about group fairness. THEMIS considers group fairness using causal analysis and uses random test generation to evaluate fairness [8]. A recent framework namely FairRec is dedicated to uncovering disadvantaged groups in recommender systems [35].

While uncovering fairness issues through testing, researchers also work to fix them. Pre-processing methods address dataset bias by correcting labels [47, 94], revising attributes [24, 48], generating non-discriminatory data [73, 91], and creating fair data representations [14, 53]. In-processing and post-processing methods reduce algorithmic bias by optimizing fairness metrics [93], using fairness-driven generative adversarial frameworks [6, 30, 91], or changing biased model outputs [39, 70]. In the SE community, retraining based on discriminatory instances is common [23, 95, 96]. Additionally, Sun et al. use causal analysis to find biased neurons [79], and Gao et al. use path analysis to select neurons for accuracy and fairness optimization [29, 31].

Discussion. To the best of our knowledge, none of the current repairing methods have been designed to consider addressing the fairness issues of model compression typified by pruning. We aim to raise awareness of fairness issues in model compression, which is a critical step towards the development of responsible AI systems.

7 Conclusion

Inspired by ethics-aware software engineering, we propose and develop Ballot, an innovative fairness-aware deep neural network pruning framework, powered by conflict-detection-based mask generation and training refinement. It identify accurate and fair tickets (pruned model prototype) and refine the ticket training process to obtain the high-performance pruned models. Our evaluation results show that Ballot effectively mitigates pruning fairness problems and outperforms all baselines in fairness and accuracy.

8 Data Availability

All source code and data used in our work can be found at [1].

Acknowledgement

We thank the anonymous reviewers for their constructive comments. This research is supported by the National Key Research and Development Program of China (2023YFB3107400), the National Natural Science Foundation of China (62376210, 62161160337, 62132011, U21B2018, U20A20177, 62206217), the Shaanxi Province Key Industry Innovation Program (2023-ZDLGY-38, 2021ZDLGY01-02). Chao Shen is the corresponding author.

References

[1] [n. d.]. Anonymized Repository - Anonymous GitHub. https://anonymous.4open.science/r/Ballot-506E.

[2] [n. d.]. Rasbt/Deeplearning-Models: A Collection of Various Deep Learning Architectures, Models, and Tips. https://github.com/rasbt/deeplearning-models/tree/master.

[3] [n. d.]. Torch.Nn — PyTorch 2.1 Documentation. https://pytorch.org/docs/stable/nn.html#module-torch.nn.utils.

[4] 2020. CIFAR-100 Datasets. https://www.cs.toronto.edu/~kriz/cifar.html. https://www.cs.toronto.edu/~kriz/cifar.html.

[5] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*. 265–283.

[6] Tameem Adel, Isabel Valera, Zoubin Ghahramani, and Adrian Weller. 2019. One-Network Adversarial Fairness. *Proc. of AAAI* (2019), 2412–2420. https://doi.org/10.1609/aaai.v33i01.33012412

[7] Aniya Agarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2018. Automated Test Generation to Detect Individual Discrimination in AI Models. *arXiv preprint arXiv:1809.03260* (2018). arXiv:1809.03260

[8] Rico Angell, Brittany Johnson, Yuriy Brun, and Alexandra Meliou. 2018. Themis: Automatically Testing Software for Discrimination. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*. 871–875. https://doi.org/10.1145/3236024.3264590

[9] Muhammad Hilmi Asyrofi, Zhou Yang, Imam Nur Bani Yusuf, Hong Jin Kang, Ferdian Thung, and David Lo. 2021. Biasfinder: Metamorphic test generation to uncover bias for sentiment analysis systems. *IEEE Transactions on Software Engineering* (2021), 5087–5101.

[10] Fatma Başak Aydemir and Fabiano Dalpiaz. 2018. A Roadmap for Ethics-Aware Software Engineering. In *Proceedings of the International Workshop on Software Fairness*. 15–21. https://doi.org/10.1145/3194770.3194778

[11] Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? *Proc. of NeurIPS* (2014).

[12] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* (2013), 1798–1828.

[13] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. 2021. Robustness may be at odds with fairness: An empirical study on class-wise accuracy. In *NeurIPS 2020 Workshop on pre-registration in machine learning*. 325–342.

[14] Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H. Chi. 2017. Data Decisions and Theoretical Implications When Adversarially Learning Fair Representations. *arXiv:1707.00075 [cs]* (July 2017). arXiv:1707.00075 [cs]

[15] Cody Blakeney, Nathaniel Huish, Yan Yan, and Ziliang Zong. 2021. Simon says: Evaluating and mitigating bias in pruned neural networks with knowledge distillation. *arXiv preprint arXiv:2106.07849* (2021).

[16] Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. *arXiv preprint arXiv:1608.08868* (2016).

[17] Tim Brennan and William L. Oliver. 2013. Emergence of Machine Learning Techniques in Criminology: Implications of Complexity in Our Data and in Research Questions. *Criminology & Pub. Pol'y* (2013), 551.

[18] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models Are Few-Shot Learners. https://doi.org/10.48550/arXiv.2005.14165 arXiv:2005.14165 [cs]

[19] Yuriy Brun and Alexandra Meliou. 2018. Software Fairness. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 754–759. https://doi.org/10.1145/3236024.3264838

[20] T. Anne Cleary. 1966. Test Bias: Validity of the Scholastic Aptitude Test for Negro and White Students in Integrated Colleges. *ETS Research Bulletin Series* (1966), i–23.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[22] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through Awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on - ITCS '12*. 214–226. https://doi.org/10.1145/2090236.2090255

[23] Ming Fan, Wenying Wei, Wuxia Jin, Zijiang Yang, and Ting Liu. 2022. Explanation-guided fairness testing through genetic algorithm. In *Proc. of ICSE*. 871–882.

[24] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and Removing Disparate Impact. In *Proc. of KDD*. 259–268. https://doi.org/10.1145/2783258.2783311

[25] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[26] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Proc. of ICLR*.

[27] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *Proc. of ICML*. 3259–3269.

[28] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness Testing: Testing Software for Discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 498–510. https://doi.org/10.1145/3106237.3106277

[29] Xuanqi Gao. 2022. FairNeuron.

[30] Xuanqi Gao, Juan Zhai, Shiqing Ma, Chao Shen, Yufei Chen, and Qian Wang. 2022. Fairneuron: Improving Deep Neural Network Fairness with Adversary Games on Selective Neurons. In *Proc. of ICSE*. 921–933. https://doi.org/10.1145/3510003.3510087

[31] Xuanqi Gao, Juan Zhai, Shiqing Ma, Chao Shen, Yufei Chen, and Shiwei Wang. 2023. CILIATE: Towards Fairer Class-based Incremental Learning by Dataset and Training Refinement. *arXiv preprint arXiv:2304.04222* (2023).

[32] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. 2020. Shortcut Learning in Deep Neural Networks. *Nature Machine Intelligence* 2, 11 (November 2020), 665–673. https://doi.org/10.1038/s42256-020-00257-z arXiv:2004.07780 [cs, q-bio]

[33] Robert M. Guion. 1966. Employment Tests and Discriminatory Hiring. *Industrial Relations: A Journal of Economy and Society* (1966), 20–37.

[34] Antonio Gulli and Sujit Pal. 2017. *Deep learning with Keras*. Packt Publishing Ltd.

[35] Huizhong Guo, Jinfeng Li, Jingyi Wang, Xiangyu Liu, Dongxia Wang, Zehong Hu, Rong Zhang, and Hui Xue. 2023. FairRec: Fairness Testing for Deep Recommender Systems. *arXiv preprint arXiv:2304.07030* (2023).

[36] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[37] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[38] Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Proc. of NeurIPS*. 1135–1143.

[39] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of Opportunity in Supervised Learning. *Proc. of NeurIPS* (2016), 3315–3323.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*. 770–778.

[41] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proc. of CVPR*. 4340–4349.

[42] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[43] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Technical Report.

[44] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proc. of CVPR*. 2704–2713.

[45] Weipeng Jiang, Chao Shen, Chenhao Lin, Jingyi Wang, Jun Sun, and Xuanqi Gao. 2023. Black-Box Fairness Testing with Shadow Models. In *International Conference on Information and Communications Security*. Springer, 467–484.

[46] Tian Jin. 2022. *On neural network pruning's effect on generalization*. Ph.D. Dissertation.

[47] Faisal Kamiran and Toon Calders. 2009. Classifying without Discriminating. In *2009 2nd International Conference on Computer, Control and Communication*. 1–6.

[48] Faisal Kamiran and Toon Calders. 2012. Data Preprocessing Techniques for Classification without Discrimination. *Knowledge and Information Systems* (2012), 1–33. https://doi.org/10.1007/s10115-011-0463-8

[49] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. 2009. Attribute and simile classifiers for face verification. In *Proc. of ICCV*. 365–372.

[50] Ya Le and Xuan Yang. 2015. Tiny imagenet visual recognition challenge. *CS 231N* (2015), 3.

[51] Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Proc. of NeurIPS* (1989).

[52] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).

[53] Yanhui Li, Linghan Meng, Lin Chen, Li Yu, Di Wu, Yuming Zhou, and Baowen Xu. [n. d.]. Training Data Debugging for the Fairness of Machine Learning Software. In *Proc. of ICSE*. 2215–2227. https://doi.org/10.1145/3510003.3510091

[54] Zhangheng LI, Tianlong Chen, Linyi Li, Bo Li, and Zhangyang Wang. 2023. Can Pruning Improve Certified Robustness of Neural Networks? *Transactions on Machine Learning Research* (2023).

[55] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. 2021. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* (2021), 370–403.

[56] Sicong Liu, Yingyan Lin, Zimu Zhou, Kaiming Nan, Hui Liu, and Junzhao Du. 2018. On-Demand Deep Model Compression for Mobile Devices: A Usage-Driven Model Selection Framework. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 389–400. https://doi.org/10.1145/3210240.3210337

[57] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proc. of ICCV*.

[58] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. 2022. Estimating the carbon footprint of bloom, a 176b parameter language model. *arXiv preprint arXiv:2211.02001* (2022).

[59] Jian-Hao Luo and Jianxin Wu. 2017. An entropy-based pruning method for cnn compression. *arXiv preprint arXiv:1706.05791* (2017).

[60] Xiaolong Ma, Geng Yuan, Xuan Shen, Tianlong Chen, Xuxi Chen, Xiaohan Chen, Ning Liu, Minghai Qin, Sijia Liu, Zhangyang Wang, et al. 2021. Sanity checks for lottery tickets: Does your winning ticket really win the jackpot? *Proc. of NeurIPS* (2021), 12749–12760.

[61] Xiaolong Ma, Geng Yuan, Xuan Shen, Tianlong Chen, Xuxi Chen, Xiaohan Chen, Ning Liu, Minghai Qin, Sijia Liu, Zhangyang Wang, and Yanzhi Wang. 2021. Sanity Checks for Lottery Tickets: Does Your Winning Ticket Really Win the Jackpot?. In *Proc. of NeurIPS*. 12749–12760.

[62] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, and Stefan Wagner. 2022. Software Engineering for AI-Based Systems: A Survey. *ACM Transactions on Software Engineering and Methodology* (2022), 37e:1–37e:59. https://doi.org/10.1145/3487043

[63] Rahul Mishra, Hari Prabhat Gupta, and Tanima Dutta. 2020. A Survey on Deep Neural Network Compression: Challenges, Overview, and Solutions. arXiv:2010.03954 [cs, eess]

[64] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proc. of CVPR*. 11264–11272.

[65] Kaiming Nan, Sicong Liu, Junzhao Du, and Hui Liu. 2019. Deep Model Compression for Mobile Platforms: A Survey. *Tsinghua Science and Technology* (2019), 677–693. https://doi.org/10.26599/TST.2018.9010103

[66] Renkun Ni, Hong-min Chu, Oscar Castañeda, Ping-yeh Chiang, Christoph Studer, and Tom Goldstein. 2020. Wrapnet: Neural net inference with ultra-low-resolution arithmetic. *arXiv preprint arXiv:2007.13242* (2020).

[67] Cathy O'neil. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy.* Crown.

[68] PaddlePaddle. [n. d.]. prune model-API Document.

[69] Michela Paganini. 2020. Prune responsibly. *arXiv preprint arXiv:2009.09936* (2020).

[70] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. [n. d.]. On Fairness and Calibration. ([n. d.]), 10.

[71] Adam Polyak and Lior Wolf. 2015. Channel-level acceleration of deep face representations. *IEEE Access* (2015), 2163–2175.

[72] Manish Raghavan, Solon Barocas, Jon Kleinberg, and Karen Levy. 2020. Mitigating Bias in Algorithmic Hiring: Evaluating Claims and Practices. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 469–481.

[73] Prasanna Sattigeri, Samuel C. Hoffman, Vijil Chenthamarakshan, and Kush R. Varshney. 2019. Fairness GAN: Generating Datasets with Fairness Properties Using a Generative Adversarial Network. *IBM Journal of Research and Development* (2019), 3–1.

[74] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM* (2020), 54–63.

[75] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proc. of ICCV*. 618–626.

[76] Agam Shah. 2023. Nvidia CEO Huang: Get Ready for Software 3.0.

[77] Jieke Shi, Zhou Yang, Bowen Xu, Hong Jin Kang, and David Lo. 2022. Compressing pre-trained models of code into 3 mb. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–12.

[78] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[79] Bing Sun, Jun Sun, Long H Pham, and Jie Shi. 2022. Causality-based neural network repair. In *Proc. of ICSE*. 338–349.

[80] Shyam A Tailor, Javier Fernandez-Marques, and Nicholas D Lane. 2020. Degree-quant: Quantization-aware training for graph neural networks. *arXiv preprint arXiv:2008.05000* (2020).

[81] Pengwei Tang, Wei Yao, Zhicong Li, and Yong Liu. 2023. Fair Scratch Tickets: Finding Fair Sparse Networks Without Weight Training. In *Proc. of CVPR*. 24406–24416.

[82] TensorflowBlog. [n. d.]. TensorFlow Model Optimization Toolkit — Pruning API.

[83] Huan Tian, Tianqing Zhu, Wei Liu, and Wanlei Zhou. 2022. Image fairness in deep learning: problems, models, and challenges. *Neural Computing and Applications* (2022), 12875–12893.

[84] Qi Tian, Kun Kuang, Kelu Jiang, Fei Wu, and Yisen Wang. 2021. Analysis and Applications of Class-Wise Robustness in Adversarial Training. In *Proc. of KDD*. 1561–1570.

[85] PyTorch Tutorials. [n. d.]. Pruning Tutorial.

[86] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. 2018. Automated Directed Fairness Testing. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE 2018*. 98–108. https://doi.org/10.1145/3238147.3238165

[87] Elmira van den Broek, Anastasia Sergeeva, and Marleen Huysman. 2019. Hiring Algorithms: An Ethnography of Fairness in Practice. (2019).

[88] Ana Ware. 2022. How Giant AI Workloads and the Looming "Bandwidth Wall" Are Impacting System Architectures.

[89] Yawen Wu, Dewen Zeng, Xiaowei Xu, Yiyu Shi, and Jingtong Hu. 2022. Fairprune: Achieving fairness through pruning for dermatological disease diagnosis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 743–753.

[90] Xiaofei Xie, Lei Ma, Haijun Wang, Yuekang Li, Yang Liu, and Xiaohong Li. 2019. Diffchaser: Detecting disagreements for deep neural networks.

[91] Depeng Xu, Shuhan Yuan, Lu Zhang, and Xintao Wu. 2018. Fairgan: Fairness-aware Generative Adversarial Networks. In *2018 IEEE International Conference on Big Data (Big Data)*. 570–575.

[92] Zhou Yang, Muhammad Hilmi Asyrofi, and David Lo. 2021. Biasrv: Uncovering biased sentiment predictions at runtime. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1540–1544.

[93] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating Unwanted Biases with Adversarial Learning. In *Proc. of AAAI*. 335–340. https://doi.org/10.1145/3278721.3278779

[94] Lu Zhang, Yongkai Wu, and Xintao Wu. 2017. Achieving Non-Discrimination in Data Release. In *Proc. of KDD*. 1335–1344. https://doi.org/10.1145/3097983.3098167

[95] Lingfeng Zhang, Yueling Zhang, and Min Zhang. 2021. Efficient white-box fairness testing through gradient search. In *Proc. of ISSTA*. 103–114.

[96] Peixin Zhang, Jingyi Wang, Jun Sun, Guoliang Dong, Xinyu Wang, Xingen Wang, Jin Song Dong, and Ting Dai. 2020. White-Box Fairness Testing through Adversarial Sampling. In *Proc. of ICSE*. 949–960. https://doi.org/10.1145/3377811.3380331

[97] Yedi Zhang, Zhe Zhao, Guangke Chen, Fu Song, Min Zhang, Taolue Chen, and Jun Sun. 2022. QVIP: an ILP-based formal verification approach for quantized neural networks. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–13.

[98] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. https://doi.org/10.48550/arXiv.2303.18223 arXiv:2303.18223 [cs]

[99] Jie Zhu, Leye Wang, and Xiao Han. 2022. Safety and Performance, Why not Both? Bi-Objective Optimized Model Compression toward AI Software Deployment. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–13.

Received 2024-04-12; accepted 2024-07-03